Enforcing Browser Anonymity with Quantitative Information Flow

Frédéric Besson, Nataliia Bielova, Thomas Jensen

Enforcing Browser Anonymity with Quantitative Information Flow

Frédéric Besson, Nataliia Bielova, Thomas Jensen

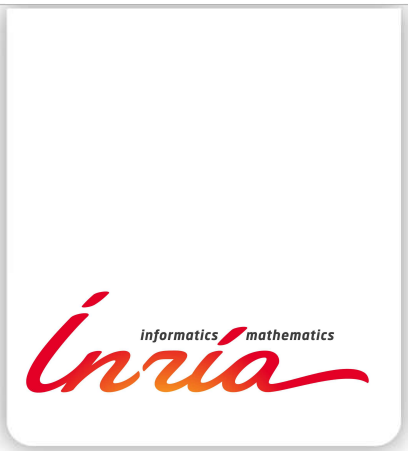
RESEARCH REPORT

N° 8532

Mai 2014

Project-Teams Celtique, Indes

ISSN 0249-6399 ISRN INRIA/RR--8532--FR+ENG



Enforcing Browser Anonymity with Quantitative Information Flow

Frédéric Besson, Nataliia Bielova, Thomas Jensen

Project-Teams Celtique, Indes

Research Report n° 8532 — Mai 2014 — 30 pages

Abstract: Web tracking companies use device fingerprinting to distinguish the users of the websites by checking the numerous properties of their machines and web browsers. One way to protect the users' privacy is to make them switch between different machine and browser configurations. We propose a formalization of this privacy enforcement mechanism. We use information-theoretic channels to model the knowledge of the tracker and the fingerprinting program, and show how to synthesise a randomisation mechanism that defines the distribution of configurations for each user. This mechanism provides a strong guarantee of *privacy* (the probability of identifying the user is bounded by a given threshold) while maximising *usability* (the user switches to other configurations rarely). To find an optimal solution, we express the enforcement problem of randomization by a linear program. We investigate and compare several approaches to randomization and find that more efficient privacy enforcement would often provide lower usability. Finally, we relax the requirement of knowing the fingerprinting program in advance, by proposing a randomisation mechanism that guarantees privacy for an arbitrary program.

Key-words: privacy enforcement, web tracking countermeasure, quantitative information flow

RESEARCH CENTRE
RENNES – BRETAGNE ATLANTIQUE

Campus universitaire de Beaulieu
35042 Rennes Cedex

Assurer l'anonymat dans un navigateur Web par quantification des flots d'information

Résumé : Pour traquer les utilisateurs de site web, certaines entreprises utilisent une technique basée sur la prise d'empreintes. Celle-ci consiste à distinguer les utilisateurs en collectant de nombreuses caractéristiques de leur machine ou navigateur. Intuitivement, il est possible de se protéger contre la prise d'empreinte en changeant régulièrement de machine ou de configuration de navigateur. Nous proposons une formalisation de système de protection de la vie privée. Nous utilisons la notion de canal (*channel*) pour modéliser à la fois la connaissance de l'attaquant et le programme de prise d'empreinte. À partir de ce modèle, nous montrons comment randomiser la distribution des utilisateurs. Nous apportons une forte garantie quant au respect de la vie privée (la probabilité d'identifier un utilisateur est bornée par un seuil donné) tout en maximisant l'ergonomie (l'utilisateur change de configuration aussi rarement que possible). Pour synthéthiser une randomisation optimale, nous exprimons le problème comme un program linéaire. Nous avons examiné différentes approches pour randomiser et nous avons constaté que les algorithmes les plus efficaces sont responsables d'une moindre ergonomie. Finalement, nous proposons un mécanisme de randomisation qui garantis l'anomymat pour un programme arbitraire.

Mots-clés : privacy enforcement, web tracking countermeasure, quantitative information flow

name	p(name)
`Firefox`	0.49
`Chrome`	0.49
`Opera`	0.02

```
1 if (name = "Opera")
2   then o := A;
3 else o := B;
4 output o;
```

Figure 1: A priori distribution of names and a program `P1`.

1 Introduction

Web tracking companies are actively using device fingerprinting to identify the users of the websites by checking the numerous properties of their machines and web browsers. While this technique is of great value to trackers, it clearly violates users privacy. The Panopticlick project [12] was the first to demonstrate the power of fingerprinting, while recent research shows that this technique is widely used by web tracking companies [1, 22].

Today, only few solutions exist for protecting the users from being fingerprinted. Acar *et al.* [1] have analysed these solutions and concluded that none of them can guarantee user privacy. For example, the Firegloves [4] browser extension returns randomized values when queried for certain browser attributes. However since the same attributes can be accessed via different browser APIs, the users of Firegloves become more uniquely identifiable than users who do not install this extension. Nevertheless, the idea of such randomisation is a promising approach to counter fingerprinting but its foundations should be developed further. In this paper, we propose a theory of privacy enforcement by randomisation and show *what privacy guarantee can be achieved*. From this theory, we derive an enforcement mechanism for obtaining this guarantee.

Fingerprinting is based on the fact that the user identity strongly relates to its browser properties, over which a script operates while collecting this data. A standard way to define how much information is leaked from a secret input to the program to the observable output, is by quantitative information flow (QIF) definitions of information leakage. We present a simple example of a possibly fingerprinting program and try to apply well-known QIF definitions of information leakage to discover what privacy guarantee we can obtain.

Example 1. For a simple illustration, consider the distribution of the browser names and the potentially fingerprinting program `P1` from Fig. 1. The distribution of browser names is known to the tracker and is called an *a priori distribution*. Assuming there are 50 visitors to the website, only one will have an Opera browser, and hence will be uniquely identified by executing the program. Notice that other 49 visitors are indistinguishable since the execution of the program will yield an output `o = B` for all of them.

1.1 QIF definitions as a measure of anonymity

To compute how much information a tracker learns about the user's browser by executing the program, we can use the standard definitions of quantitative information flow (i.e., Shannon entropy, min-entropy, guessing entropy, etc.) and compute an information leakage averaged for all possible outputs represented by a number of bits. Here we propose to compare related definitions based on the probability of learning the program secret input.

QIF definitions are based on the fact that a program output transforms an *a priori distribution* of secrets into an *a posteriori* distribution of secrets given an observed output. In Fig. 2 we show two a posteriori distributions that represent the knowledge of the tracker corresponding to two possible program outputs `o = A` and `o = B`. Each QIF definition uses a different metric to

name	$p(\text{name}\|\texttt{A})$
`Firefox`	0
`Chrome`	0
`Opera`	1

name	$p(\text{name}\|\texttt{B})$
`Firefox`	0.5
`Chrome`	0.5
`Opera`	0

Figure 2: A posteriori distribution of names for `o=A` and `o=B`.

evaluate how the attacker's knowledge has changed after running the program and observing an output.

Smith [25] proposes a metric of *vulnerability* that is later used in a min-entropy definition of QIF. Vulnerability $V(i|o)$ computes the maximum probability of identifying the secret i given an output o[1]. To evaluate the vulnerability of the program, Smith proposes to compute an average of all possible outputs:

$$\begin{array}{lcl} V(\text{name}|\texttt{A}) & = & \max\{0, 0, p(\texttt{Opera}|\texttt{A})\} = 1 \\ V(\text{name}|\texttt{B}) & = & \max\{p(\texttt{Firefox}|\texttt{B}), p(\texttt{Chrome}|\texttt{B})\} = 0.5 \\ V(\texttt{P}) & = & p(\texttt{A}) \cdot V(\text{name}|A) + p(\texttt{B}) \cdot V(\text{name}|B) \\ & = & 0.02 \cdot 1 + 0.98 \cdot 0.5 = 0.51 \end{array}$$

Vulnerability $V(\texttt{P})$ means that on average, given an output, the probability of guessing the secret after running a program `P` is 0.51. However, the secret name `Opera` can be leaked completely by output `A`, even though this output has a low probability ($p(\texttt{A}) = 0.02$). Moreover, the leakage of exactly this output makes the user uniquely identifiable given there are 50 visitors to the website. Therefore, approaches that evaluate min-entropy of the program are not suitable for the enforcement of anonymity.

Clarkson *et al.* [7] propose a notion of attacker's *belief* about possible values of a secret, represented as an a priori probability distribution (pre-belief). This belief can be revised by running the program with the actual secret, thus obtaining an a posteriori distribution (post-belief). Information leakage is defined as a difference in distance between a pre-belief and the actual secret, and a post-belief and the secret. This definition of leakage is shown to be equivalent to the *self-information* of an output corresponding to a secret and is measured in bits. For our program `P`, we can compute a leakage separately for Opera browser users (leakage of output `A`) and for Chrome and Firefox users (leakage of output `B`):

$$\begin{array}{ll} I(\texttt{A}) & = -\log_2 p(\texttt{A}) = -\log_2 0.02 = 5.64 \text{ bits} \\ I(\texttt{B}) & = -\log_2 p(\texttt{B}) = -\log_2 0.98 = 0.03 \text{ bits} \end{array}$$

The goal of these previous works was to evaluate the information learnt about the secret program input, and was not related to user identifiability. However, since the secret program input is related to the user identity, the amount of information leaked is also closely related to the identifiability of the user. For example, Firefox or Chrome user would leak $-\log_2 0.98$ bits of information, in other words, the probability of her browser name is 0.98, which makes her one in $0.98 * 50 = 49$ other indistinguishable users. On the contrary, Opera user would leak $-\log_2 0.02$ bits, that makes her to be one in $0.02 * 50 = 1$ users, therefore such user is uniquely identified.

In our previous work [2], we propose a hybrid monitor that evaluates an amount of information leaked for each secret input (i.e., for each program execution). The definition of leakage is *self-information*, that coincides with Clarkson *et al.*'s belief-based definition. We could apply a naive counter-measure against user identification by stopping executions which leak a quantity

[1] Whenever an output is impossible for a given input, the probability is 0, *e.g.*, $p(\texttt{Firefox}|\texttt{A}) = 0$

of information above a given threshold. However, such an approach is not sufficient to enforce privacy in a presence of a tracker that can observe the absence of output. For example, putting a threshold of 1 bit would mean that a user should be one in a set of at least $\frac{1}{2^1} * 50 = 25$ users. Yet, the naive counter-measure would only stop the execution for Opera users. Therefore, by observing non-termination, the tracker would identify with certainty Opera users and thus violate their privacy.

Inspired by the Clarkson *et al.*'s work on belief revision, Mardziel *et al.* [20, 21] propose a definition of *knowledge threshold security* stating that a program is secure if all the post-beliefs of all possible secrets are bounded by some threshold t. In case of a fingerprinting program example, this definition would mean that for all secret names and outputs o: $p(\text{name}|o) \leq t$ holds. This definition can also be rewritten using the notion of vulnerability: for our program `P`, we have

$$\max\{V(\text{name}|\texttt{A}), V(\text{name}|\texttt{B})\} \leq t.$$

This definition provides a stronger notion of input secrecy than the definition of vulnerability. Espinoza and Smith [13] discuss this definition and name it *worst-case posterior vulnerability* underlining that it is very biased towards the worst output. Indeed, we would get the same worst-case posterior vulnerability for a program `output name;` that always leaks the secret completely. Nevertheless, this definition provides a very strong guarantee with respect to the probability that the secret will be guessed after an execution of the program. In order to enforce *threshold based security*, Mardziel *et al.* [20,21] suggest to run the program if the threshold holds for all the values of the secret input and not to run it in case there is at least one value for which the guarantee does not hold. This radical approach does not allow all the safe users to run the program. In our example it would mean that thanks to 1 user of an Opera browser, 49 other website visitors will not execute the program and probably would not get the service they wanted when they visited the website.

In this paper, we will define a strong definition of privacy based on the worst-case posterior vulnerability, and adapt it such that it evaluates the *probability of guessing the user identity* instead of the probability of guessing the secret program input. For example, given 50 website visitors, program `P` will be evaluated to provide a probability of guessing the identity equal to 1 for Opera users and $\frac{1}{49}$ for Firefox and Chrome users. Then, for a threshold $t = \frac{1}{25}$, we will provide a mechanism that randomizes the browser name for Opera users, and not influence the experience of Firefox and Chrome users.

1.2 Attacker model and assumptions

We consider terminating (deterministic or probabilistic) programs operating over a finite range of inputs. Upon termination, the program returns a single value. As we only consider terminating programs, termination is observable by the attacker and the security property we guarantee is therefore termination sensitive. In our model, the attacker has arbitrary computing power, he provides the program and observes the output. The input of the program is secret and represents a browser configuration. However, the attacker has perfect knowledge over the distribution of the secret inputs. Our enforcement works by randomising the program inputs, and the attacker has access to the precise description of the randomisation mechanism.

1.3 Organisation of the paper

In Section 2, we present background information about information theory. We shall be using an information-theoretic model in which programs as well as user characteristics are modelled as channels $\mathcal{P}$ and $\mathcal{U}$. In Section 3, we explain how fingerprinting protection can be modelled in

terms of channels. The deterministic channel $\mathcal{U}$ models the distribution of the browser statistics and maps identities to browser configurations. The channel $\mathcal{P}$ models the fingerprinting script. The composition of channels $\mathcal{U} \otimes \mathcal{P}$ models sequential composition. In Section 4, we explain our t-privacy guarantee that consists in bounding the worst-case probability $\mathbb{P}$ of guessing an identity based on the observation of a program output.

$$\mathbb{P}(\mathcal{U} \otimes \mathcal{P}) \leq t. \tag{1}$$

We propose enforcement mechanisms for t-privacy, which are based on lying about your browser configurations. Formally, enforcement consists in synthesising a channel $\mathcal{R}$ such that $\mathcal{R} \otimes P$ is t-private while optimising the quantity $\mathbb{U}(\mathcal{R},\mathcal{U})$ which measures the amount of randomisation of $\mathcal{R}$ w.r.t $\mathcal{U}$.

In Section 5, we show how finding this channel $\mathcal{R}$ can be reduced to solving a *linear program* where the variables are the entries of the $\mathcal{R}$ matrix and the constraints come from the privacy guarantee, while the objective function models usability *i.e.*, $\mathbb{U}(\mathcal{R},\mathcal{U})$. In Section 6, we explore three kinds of enforcement algorithms derived from this linear programming encoding. A purely static enforcement would solve the linear program but the complexity of the problem is likely to make this approach infeasible. Instead we propose to look for locally safe solutions that will work for individual users but will not solve the global problem optimally. We define the notion of locally safe and propose a modified algorithm for its enforcement. This algorithm has the advantage of being decentralised and can furthermore be adapted to be run in an incremental fashion.

Finally, in Section 7 we consider a generalisation of the enforcement problem in which the enforcement has to work for all programs, and not just one given program. We show how to construct a randomised user channel $\mathcal{R}$ from a given user channel $\mathcal{U}$, that enforces t-privacy for any program channel: $\forall \mathcal{P} : \mathbb{P}(\mathcal{R} \otimes \mathcal{P}) \leq t$.

Due to the lack of space, the proofs of the theorems of this paper can be found in the Appendix.

1.4 Contributions

The main contributions can be summarised as:

- A model of the problem of privacy protection against fingerprinting programs, based on information-theoretic channels representing the statistics of browser properties and the program.
- A novel definition of privacy for such systems, that ensures that the probability of an attacker identifying a user is bounded by a given threshold. We show that the enforcement of privacy can be achieved by randomizing the browser properties, and that this randomization problem can be reduced to solving linear programs.
- Three algorithms (a purely static, a purely dynamic and an efficient incremental algorithm) for enforcing privacy against a particular finger-printing program. All algorithms ensure the strong privacy guarantee, *i.e.*, that the probability of being identified is smaller than a given threshold. The algorithms optimize the solution with respect to additional "usability" constraints, which ensure that randomization is used as little as possible.
- A general result about how user privacy can be guaranteed for any program that the user might run. This represents the worst case scenario, in which a program checks all the possible browser properties of the user. This result is important in the case where it

is difficult or impossible to construct the information-theoretic channel that models the program (*e.g.*, due to the dynamicity of a language such as JavaScript).

2 Background on information theory

We follow the standard approach and model a system using an information-theoretic channel [8]. In this section we present several most related definitions from this field, such as the probability of error [5, 6] and vulnerability [25]. We then present a cascade of channels [13] that we use to model our system and finish the section with several definitions that we will use in our modeling.

A *channel* is a triple $\mathcal{C} = (I, O, C)$, where I and O are finite sets of inputs and outputs, and C is a *channel matrix* of size $|I| \times |O|$, where each cell is between 0 and 1, and each row sums up to 1. Each cell $C[i, o]$ is the probability of getting an output o given an input i, and we also denote $C[i, o]$ by a conditional probability $p(o|i)$. A channel $\mathcal{C}$ is *deterministic* if each cell of C is either 0 or 1, meaning that each input can produce only one output.

For a deterministic channel $\mathcal{C}$, we write $Im(C, i)$ for the unique o such that $C[i, o] = 1$ and $Pre(C, o)$ for the set of inputs that can produce o:

$$\begin{array}{lcl} Im(C,i) & = & o \text{ iff } C[i,o] = 1 \\ Pre(C,o) & = & \{i | C[i,o] = 1\}. \end{array}$$

An input of a channel is a *secret information*, and the output is *observable information*. The set of input values can be considered as a set of mutually exclusive facts, or hypotheses. A probability distribution $p(\cdot)$ over I is called *a priori probability*. For a given channel (I, O, C), it induces a probability distribution over O:

$$p(o) = \sum_{i \in I} p(i, o) = \sum_{i \in I} p(o|i) \cdot p(i).$$

When an output o is observed, the probability that an input has a certain value i is given by the conditional probability $p(i|o)$, that is called *a posterior probability* of i given o. The a posterior probability is usually different from the a priori probability because an observation o gives some evidence that changes the degree of belief in the hypothesis i. The a posteriori probability can be computed from the a priori probability using Bayes' theorem:

$$p(i|o) = \frac{p(o|i) \cdot p(i)}{p(o)}.$$

Hypothesis testing tries to infer the *true* hypothesis from the observed fact[2]. In general, it is impossible to determine the true hypothesis from just a single observation. Following the hypothesis testing approach [5], we compute the probability of error, or the probability of making the wrong guess given the observation o. The probability of error is defined as follows. An observer's attempt to guess the secret i given an observation o is expressed as a decision function $f : O \to I$. The Bayesian framework determines that the best possible decision function f_B is the function that minimises the probability of error in guessing the secret. This function is obtained by the MAP (Maximum Aposteriori Probability) criterion: it chooses an input i with maximum $p(i|o)$:

$$f_B(o) = i \Rightarrow \forall i' : p(i|o) \geq p(i'|o).$$

[2]Usually, an attacker tries to derive the value of a secret input from observing some output o.

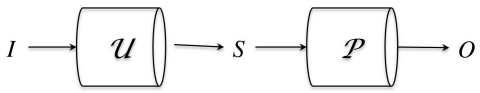


Figure 3: Cascade of channels $\mathcal{C}_0$ and $\mathcal{C}_1$

The probability of error associated with f_B is called the *Bayes risk* and is given by

$$\mathbb{P}_e^{aver} = 1 - \sum_{o \in O} p(o) \cdot \max_{i \in I} p(i|o).$$

We denote the Bayes risk with *aver* superscript because it evaluates the probability an observer makes an error averaged for all possible channel outputs. Dually, we can compute an *average probability of guessing the secret input* given an observation:

$$\mathbb{P}^{aver} = 1 - \mathbb{P}_e^{aver} = \sum_{o \in O} p(o) \cdot \max_{i \in I} p(i|o). \tag{2}$$

The metric of *vulnerability* introduced by Smith [25] coincides with the definition of the average probability of guessing. Differently from [5], Smith first introduces a vulnerability of a distribution as a worst-case probability that an observer would guess the value of a secret input in one try. After observing an output o, an a posterior distribution has the vulnerability $V(I|o) = \max_{i \in I} p(i|o)$. Then, Smith proposed to average this probability per each channel output: $V(\mathcal{C}) = \sum_{o \in O} p(o) \cdot V(I|o)$, thus obtaining the same definition.

In our modeling, we will use a classic construction of a *cascade* of channels [9], where an output of one channel is used as an input of another channel as depicted in Fig. 3. The first applications of a cascade of channels to quantitative information flow were proposed by Köpf and Smith [18], who evaluated its min-capacity, and the recent work of Espinoza and Smith [13] redefine a cascade of channels for an arbitrary a priori distribution and show the properties of vulnerability metric when applied to a cascade of channels.

Definition 1. Two channels $\mathcal{C}_1 = (I, S, C_1)$ and $\mathcal{C}_2 = (S, O, C_2)$ form a *cascade of channels* $\mathcal{C}_1 \otimes \mathcal{C}_2 = (I, O, C)$, where

$$(C_1 \otimes C_2)[i, o] = \sum_{s \in S} C_1[i, s] \cdot C_2[s, o].$$

We now formalise the sets of indistinguishable inputs given a (possibly probabilistic) channel. The following definition generalises the definition of equivalence classes from [2, 25] to the case of probabilistic channels. Two secret inputs i and i' are indistinguishable if the corresponding rows in the channel matrix are identical.

Definition 2. Given a channel $\mathcal{C} = (I, O, C)$, two inputs i and i' are *indistinguishable* (written $i \sim_{\mathcal{C}} i'$) if for all outputs o, $C[i, o] = C[i', o]$. We write $Eq_{\mathcal{C}}(i) = \{i' \mid i \sim_{\mathcal{C}} i'\}$ the set of inputs indistinguishable from i.

For deterministic channels, the number of sets of indistinguishable secrets is equal to the number of channel outputs: $|O|$. However, it is not the case for probabilistic channels (the channel contains other numbers except for 0 and 1): here the number of sets of indistinguishable secrets can exceed the number of possible outputs (e.g., consider a case when all the rows of the matrix are different whatever is the number of outputs).

3 Modeling the fingerprinting problem

This section shows how to model fingerprinting in terms of information-theoretic channels. The input to the script is the browser configuration, that is considered to be the user's secret. We model a script as a channel that takes the secret browser configuration s and produces an output o. Let S be the finite set of all possible configurations, and O the set of all possible outputs of a script.

In information theory, a program or a script, can be modelled as an information-theoretic channel (S, O, P), assuming that the channel operates over a finite set of secrets S and produces an output from a finite set O. Each matrix element $P[s, o]$ is a conditional probability of an output o given a secret configuration s. For deterministic scripts, the corresponding channel can be extracted by running the script for all the possible inputs. These inputs are the possible browser configurations whose distribution is known to the attacker. For probabilistic scripts, probabilistic sampling would construct a reliable but approximate channel. Even better, we can construct an exact channel using symbolic computations over probability distributions. For each possible input, we can evaluate the script semantics expressed as a distribution transformer [7]. Symmetrically, we can run a weakest pre-expectation calculus [14] which extends weakest pre-condition calculus to probabilistic programs. The model acquisition problem is out of the scope of this paper, and henceforth we shall just assume that a channel matrix is given.

Given a channel, we could evaluate the probability of guessing the browser configuration s given an output o. However, what we are interested in is the *probability of guessing the user identity* and not just in knowing her configuration.

The user identities are related to the browser configurations by the browser statistics. These statistics define how many users have a certain configuration. Assuming that we know which user possesses which configuration, we can model this relation by a deterministic channel (I, S, U), where I is a finite set of user identities, S is a finite set of possible browser configurations and U is a channel matrix, where

$$U[i, s] = \begin{cases} 1 & \text{if user } i \text{ has configuration } s \\ 0 & \text{otherwise.} \end{cases}$$

By construction the matrix U is deterministic, meaning that each row contains only one cell equal to 1, and the rest are 0s. In other words, $U[i, s]$ means that a user i posseses only one configuration s. Notice that an a priori distribution of secrets in our system is an a priori distribution of user identities (both for the user channel $\mathcal{U}$ and the cascade of channels $\mathcal{U} \otimes \mathcal{P}$). Since all the users are initially equally indistinguishable, we assume that the a priori distribution $p(i)$ is a *uniform distribution*.

In order to compute the probability of identifying the user with identity i given an observation o produced by a script running in her browser, we need to combine the channels (I, S, U) and (S, O, P). Since the output s of the first channel U is provided as an input to the second channel P, these channels form a *cascade of channels*.

4 Threshold-based privacy and usability

In this section we propose our definition of privacy, called *t-privacy*. It is based on a strong guarantee that we call *probability of guessing the user identity* that computes the worst-case probability of guessing among all possible observations. We explain how and under which conditions t-privacy can be enforced, and propose an additional notion of *usability* that considers the user's requirements and helps to choose the optimal enforcement of t-privacy.

4.1 Probability of guessing the user identity and t-privacy

In Equation 2 of Section 2, we presented the definition of average probability of guessing the secret $\mathbb{P}^{aver}$ derived the from Bayes risk [5], this definition coincides with channel's vulnerability [25]. $\mathbb{P}^{aver}$ represents the fact that an observer will guess the secret in one try averaged for all possible outputs. However, our goal is to provide a very strong guarantee of privacy, such as "given an observation, a tracker has a very small probability of guessing the user identity". Therefore, we propose a different notion that we call *probability of guessing*. It computes the worst case probability of guessing the secret among all possible channel outputs.

Definition 3. The *probability of guessing the input* of a channel $\mathcal{C} = (I, O, C)$ is

$$\mathbb{P}(\mathcal{C}) = \max_{i \in I, o \in O} p(i|o)$$

and in case of uniform prior distribution,

$$\mathbb{P}(\mathcal{C}) = \max_{i \in I, o \in O} \frac{C[i,o]}{\sum_{i'} C[i',o]}.$$

Note that this definition is closely related to knowledge threshold security by Mardziel *et al.* [20, 21] and worst-case posterior vulnerability by Espinoza and Smith [13].

To model the fingerprinting problem, we use a cascade of channels $\mathcal{U} \otimes \mathcal{P}$, where $\mathcal{U}$ represents the users with their configurations and $\mathcal{P}$ represents the possibly fingerprinting program. To protect the user's privacy, we propose to put a threshold on the probability of guessing the user identity when a tracker observes the program output. Our definition resembles the definition of knowledge threshold security [20] , though we use a different motivation. Intuitively, t-privacy means that every user can be identified with the probability up to t. For example, in the spirit of k-anonymity, we can express the following policy: "every user must be indistinguishable in a set of at least k other users". Our guarantee of t-privacy, where $t = \frac{1}{k}$, enforces this policy (we compare t-privacy and k-anonymity in Section 8).

Definition 4 (Threshold-based privacy)**.** A channel $\mathcal{C}$ is *t-private* if the probability of guessing the channel's input is bounded by t: $\mathbb{P}(\mathcal{C}) \leq t$.

Using the definition of a cascade of channels, we state the probability of guessing the user identity after a tracker observes an output of a program execution:

$$\mathbb{P}(\mathcal{U} \otimes \mathcal{P}) = \max_{i \in I, o \in O} \frac{\sum_{s \in S} U[i,s] \cdot P[s,o]}{\sum_{i'} \sum_{s \in S} U[i',s] \cdot P[s,o]}.$$

Example 2. Consider the program `P1` from Example 1: `if (name = "Opera") then o := A; else o := B; output o;`. Fig. 4 shows the matrix of a user channel $\mathcal{U}$ representing (simplified) browser statistics and the matrix of a channel $\mathcal{P}$ for program `P1`. When a user with identity i_5 runs the program `P`, a tracker observes an output `A` and knows that the user has Opera configuration. The channel $\mathcal{U}$ then makes the user i_5 uniquely identifiable, therefore a posteriori probability is $p(i_5|\mathtt{A}) = \frac{U[i_5,\mathtt{Opera}] \cdot P[\mathtt{Opera},\mathtt{A}]}{U[i_5,\mathtt{Opera}] \cdot P[\mathtt{Opera},\mathtt{A}]} = 1$. When an attacker observes output `B`, he concludes that the user a different configuration than Opera. The channel $\mathcal{U}$ makes all the other users indistinguishable for the attacker thanks to the program output `B`. Therefore for all these users $p(i|\mathtt{B}) = \frac{1}{4}$ since output `B` can be obtained from Firefox and Chrome configurations. We conclude that the probability of guessing the user identity is:

$$\mathbb{P}(\mathcal{U} \otimes \mathcal{P}) = \max_{i \in I, o \in O} p(i|o) = 1.$$

U	Firefox	Opera
i_1	1	0
i_2	1	0
i_3	1	0
i_4	1	0
i_5	0	1

P	A	B
Firefox	0	1
Opera	1	0

Figure 4: User channel and channel for program P1.

We can establish the lower bound on the probability of guessing. The intuition for this bound is that a channel that best protects the secret input realizes the smallest probability of guessing. One example is a channel, where all the inputs are mapped into only one output; another example is a channel, where all the inputs are uniformly distributed between all possible outputs.

Theorem 1. *For any channel $\mathcal{C} = (I, O, C)$, the probability of guessing the input is bounded:* $\mathbb{P}(\mathcal{C}) \geq \frac{1}{|I|}$.

We compare the probability of guessing the input of the first channel $\mathcal{U}$, and of a cascade of channels $\mathcal{U} \otimes \mathcal{P}$.[3] The program can make some of the secret configurations indistinguishable, thus making the tracker learn less, and therefore the probability of guessing the user identity becomes lower. The following theorem is generic for a cascade of two channels and for any prior distribution.

Theorem 2. *The probability of guessing the identity from a user channel $\mathcal{U}$ does not increase if a program channel $\mathcal{P}$ is added:* $\mathbb{P}(\mathcal{U} \otimes \mathcal{P}) \leq \mathbb{P}(\mathcal{U})$.

In order to enforce t-privacy, we propose several mechanisms that are based on a substitution of an original user channel $\mathcal{U}$ with a randomized channel $\mathcal{R}$, that defines how exactly each user should switch to a different configuration. Therefore, t-privacy is enforced if $\mathbb{P}(\mathcal{R} \otimes \mathcal{P}) \leq t$. However, not every threshold can be enforced by user channel randomization. We state our enforceability theorem as follows.

Definition 5 (Enforceable threshold)**.** A threshold t is *enforceable* for a given user channel $\mathcal{U}$ and program channel $\mathcal{P}$ if there exists a randomised channel $\mathcal{R}$, such that $\mathcal{R} \otimes \mathcal{P}$ is t-private ($\mathbb{P}(\mathcal{R} \otimes \mathcal{P}) \leq t$).

Theorem 3 (Enforceability)**.** *A threshold t is enforceable for a user channel $\mathcal{U} = (I, S, U)$ and any program channel $\mathcal{P}$ if and only if* $\frac{1}{|I|} \leq t$.

Example 3. Continuing Example 2, a threshold $t = 0.5$ can be enforced when some Firefox users switch configurations so that Opera users are less identifiable. The randomized channel $\mathcal{R}_0$ in Fig. 5 is using this strategy to enforce t-privacy.

We can graphically represent the space of possible randomization channels $\mathcal{R}$ that enforce t-privacy, where all Firefox users would get a probability x for Firefox and $1-x$ for Opera, dually Opera users would get probability y for using Opera and $1-y$ for Firefox. The probability of guessing the user identity of a channel $\mathcal{R} \otimes \mathcal{P}$ is:

$$\max\left\{\frac{x}{4x+1-y}, \frac{1-x}{4-4x+y}, \frac{1-y}{4x+1-y}, \frac{y}{4-4x+y}\right\}.$$

R_0	Firefox	Opera
i_1	0.75	0.25
i_2	0.75	0.25
i_3	0.75	0.25
i_4	0.75	0.25
i_5	0.7	0.3

$$p(i_1|\mathtt{B}) = \frac{0.75}{0.75\cdot 4+0.7} = 0.203$$
$$p(i_5|\mathtt{B}) = \frac{0.7}{0.75\cdot 4+0.7} = 0.189$$
$$p(i_1|\mathtt{A}) = \frac{0.25}{0.25\cdot 4+0.3} = 0.192$$
$$p(i_5|\mathtt{A}) = \frac{0.3}{0.25\cdot 4+0.3} = 0.231$$

Figure 5: For program P, $\mathcal{R}_0$ enforces t-privacy with $t = 0.5$.

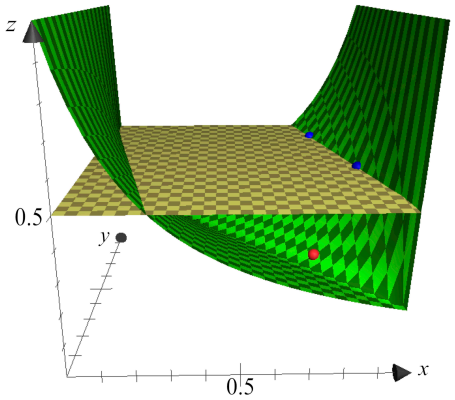


Figure 6: Probability of guessing user identity for Example 2 and bound $t = 0.5$.

In Fig. 6, the dark green surface presents a probability of guessing the user identity of a channel $\mathcal{R} \otimes \mathcal{P}$, while the yellow flat surface shows the probability of guessing the identity set to $\frac{1}{2}$. The parallelogram at the intersection of two surfaces represents the set of solutions for a randomisation channel $\mathcal{R}$. The red point $x = 0.75$, $y = 0.3$ corresponds to the randomization channel $\mathcal{R}_0$ and belongs to the surface below the threshold $t = \frac{1}{2}$.

4.2 Usability

It is important to consider usability requirements, such as "the users do not want to switch to other configurations too often since they prefer to use their original configurations as much as possible". We therefore propose a formal definition of usability.

Definition 6 (Usability)**.** Given a channel $\mathcal{U}$, the usability of a channel $\mathcal{R}$ is $\mathbb{U}(\mathcal{R},\mathcal{U}) = \sum_i R[i, Im(U, i)]$.

Usability quantifies the amount of switching of configurations incurred by a given channel $\mathcal{R}$. We aim at maximising usability *i.e.,* minimising the probability that a user needs to switch between configurations. The maximum theoretical usability is obviously obtained when $\mathcal{R} = \mathcal{U}$ and $\mathbb{U}(\mathcal{U},\mathcal{U}) = |I|$.The usability of the channel $\mathcal{R}_0$ from Fig. 5 is $\mathbb{U}(\mathcal{R}_0,\mathcal{U}) = 0.75 \cdot 4 + 0.3 = 3.3$. In the next section we show that there exists another randomised channel $\mathcal{R}_1$ that provides $\mathbb{U}(\mathcal{R}_1,\mathcal{U}) = 4$ for a theoretical optimal of 5.

[3]Our result resembles Theorem 5 from [13] proving that the min-entropy leakage of the cascade of channels is not greater than the leakage of the first link in the cascade.

5 Reduction to Linear Programming

This section shows how to construct a randomised channel $\mathcal{R}$ that ensures t-security for the composed channel $\mathcal{R} \otimes \mathcal{P}$. We parametrize the matrix of the channel $\mathcal{R}$ using the variables $R[i,s] = x_{is}$, where $i \in I, s \in S$ and define a system of constraints that such a channel must satisfy. Finding the best possible channel can then be expressed as a Linear Program.

5.0.1 Channel correctness

The channel matrix R represents conditional probabilities, therefore:

- $0 \leq x_{is} \leq 1$ for all $i \in I, s \in S$, meaning each parameter is a probability.
- $\sum_{s \in S} x_{is} = 1$ for all $i \in I$, meaning each row of the matrix represents a distribution.

5.0.2 t-privacy

: the channel $\mathcal{R}$ must guarantee t-privacy for the composed program channel $\mathcal{P}$ *i.e.*, $\mathbb{P}(\mathcal{R} \otimes \mathcal{P}) \leq t$. Expanding this expression, we get:

$$\max_{i \in I, o \in O} \frac{\sum_{s \in S} x_{is} \cdot P[s,o]}{\sum_{j \in I} \sum_{s \in S} x_{js} \cdot P[s,o]} \leq t.$$

This can be rewritten as a system of linear inequalities for all $i \in I, o \in O$:

$$\sum_{s \in S} x_{is} \cdot P[s,o] - t \cdot \sum_{j \in I} \sum_{s \in S} x_{js} \cdot P[s,o] \leq 0.$$

The system of constraints presented by the channel correctness and t-privacy requirements have a number of solutions. One solution is a channel $\mathcal{R}$ where all the matrix elements are equal to $\frac{1}{|S|}$. To see this, observe that 1) the channel is correct : $\sum_{s \in S} \frac{1}{|S|} = 1$; 2) t-privacy is guaranteed for any $\mathcal{P}$ since $\mathbb{P}(\mathcal{R} \otimes \mathcal{P}) \leq \mathbb{P}(\mathcal{R})$ (Theorem 2), and $\mathbb{P}(\mathcal{R}) = \frac{1/|S|}{\sum_{j \in I} 1/|S|} = \frac{1}{|I|} \leq t$ for any enforceable t (Theorem 3). However, this solution might not guarantee the best usability: it forces each user to use other configurations as often as his original configuration.

This last observation motivates a third requirement:

5.0.3 Usability

Our usability requirement exactly describes the desire of users to switch to other configurations as rarely as possible. Therefore, the usability of a randomised channel $\mathcal{R}$ must be maximised. Remember that usability $\mathbb{U}(\mathcal{R}, \mathcal{U}) = \sum_{i \in I} R[i, Im(U,i)]$ represents a sum of all cells in R where $U[i,s] = 1$. Therefore, we can rewrite it as a requirement to maximise the function $\sum_{(i \in I): U[i,s]=1} x_{is}$.

Combining the requirements presented in this section, we can state our problem as a *Linear Program*, where the usability must be maximised while the channel correctness and t-privacy constraints are satisfied:

$$max \ \sum_{(i \in I): U[i,s]=1} x_{is} \text{ s.t.}$$
$$\begin{cases} 0 \leq x_{is} \leq 1 & \forall i \in I, s \in S \\ \sum_{s \in S} x_{is} = 1 & \forall i \in I \\ \sum_{s \in S} x_{is} \cdot P[s,o] - t \cdot & \sum_{j \in I} \sum_{s \in S} x_{js} \cdot P[s,o] \leq 0 \\ & \forall i \in I, o \in O \end{cases}$$

U	Firefox	Opera
i_0	1	0
i_1	1	0
i_2	1	0
i_3	1	0
i_4	0	1

R	Firefox	Opera
i_0	x_{00}	x_{01}
i_1	x_{10}	x_{11}
i_2	x_{20}	x_{21}
i_3	x_{30}	x_{31}
i_4	x_{40}	x_{41}

Figure 7: Original user channel matrix from Ex. 2 and a corresponding randomised channel matrix.

R_1	Firefox	Opera
i_0	0.9	0.1
i_1	0.8	0.2
i_2	0.7	0.3
i_3	0.6	0.4
i_4	0	1

R_2	Firefox	Opera
i_0	1	0
i_1	1	0
i_2	1	0
i_3	0.5	0.5
i_4	0.5	0.5

Figure 8: Two optimal solutions for channel $\mathcal{R}$.

Example 4. We present the system of constraints for the program `P1` from Example 2. For a user channel $\mathcal{U} = (U, I, S)$, the randomised channel $\mathcal{R}$ has a matrix of $|I| \times |S|$ parameters denoted by x_{is} (see Fig. 7). The usability of $\mathcal{R}$ is computed as follows:

$$\mathbb{U}(\mathcal{R},\mathcal{U}) = \sum_{(i\in I):U[i,s]=1} x_{is} = x_{00}+x_{10}+x_{20}+x_{30}+x_{41}.$$

The constraints imposed by channel correctness are straightforward to write down. Here, we present the constraints provided by the t- privacy requirement, where the threshold $t = \frac{1}{2}$:

$$\begin{aligned}
&+\tfrac{1}{2}\,x_{00}-\tfrac{1}{2}\,x_{10}-\tfrac{1}{2}\,x_{20}-\tfrac{1}{2}\,x_{30}-\tfrac{1}{2}\,x_{40}\leq 0\\
&-\tfrac{1}{2}\,x_{00}+\tfrac{1}{2}\,x_{10}-\tfrac{1}{2}\,x_{20}-\tfrac{1}{2}\,x_{30}-\tfrac{1}{2}\,x_{40}\leq 0\\
&\ldots\\
&-\tfrac{1}{2}\,x_{01}-\tfrac{1}{2}\,x_{11}-\tfrac{1}{2}\,x_{21}+\tfrac{1}{2}\,x_{31}-\tfrac{1}{2}\,x_{41}\leq 0\\
&-\tfrac{1}{2}\,x_{01}-\tfrac{1}{2}\,x_{11}-\tfrac{1}{2}\,x_{21}-\tfrac{1}{2}\,x_{31}+\tfrac{1}{2}\,x_{41}\leq 0
\end{aligned}$$

This Linear Program has several solutions. In Fig. 8 we present two optimal solutions $\mathcal{R}_1$ and $\mathcal{R}_2$.Both solutions are optimal, and $\mathbb{U}(\mathcal{R}_1,\mathcal{U}) = \mathbb{U}(\mathcal{R}_2,\mathcal{U}) = 4$.

5.1 Feasibility

In practice, switching to a particular configuration might not be technically feasible. For instance, faithfully emulating a foreign OS might be considered too drastic a change. Other switchings of configurations could hinder an optimal user experience. For instance, switching to an arbitrary language could have undesirable side-effects. For personal reasons, certain identities might also wish to keep their configuration unaltered. To model those feasibility constraints, we can introduce a relation $\mathit{Imp} \subseteq I \times S$ representing the configuration switches that are considered impossible. The fact that those switchings between configurations might be impossible can be encoded into the Linear Program in a straightforward way. Each pair $(i, s) \in \mathit{Imp}$ yields the

constraint $x_{is} = 0$. These constraints can be used to pre-process the problem and substitute x_{is} for 0, thus reducing the number of variables of the problem.

6 Enforcement algorithms

6.1 Static approach

Linear programs can be solved in polynomial time using interior points methods. Despite a worst-case exponential complexity, the Simplex algorithm is usually superior and practice shows that, on average, only a linear number of pivot steps are needed. Furthermore, for probabilistic models, this average linear bound can be formally established [24]. A linear program can be rewritten in standard form

$$max\ c \cdot x \text{ s.t.} \begin{cases} A \cdot x \leq b \\ x \geq 0 \end{cases}$$

where A is a n, m-matrix, b and c are vectors of size m. We write $CLP(m, n, s)$ for the complexity of solving a linear program with m variables and n constraints where s is the bit-size of the matrix A *i.e.,*

$$s = \|A\| = \sum_{i,j} \lceil \log_2(|A[i,j]|) \rceil.$$

Under the condition that the matrix P is dense and the threshold t is such that t^{-1} is integer, constructing the channel $\mathcal{R}$ with optimum utility can be done in complexity $CLP(m, n, s)$ where

$$\begin{array}{rcl} m & = & |I| \cdot |S| \\ n & = & |I| \cdot |O| + |I| \cdot |S| + |I| \\ s & = & |I|^2 \cdot \|P\| + |I| \cdot \|P\| + |I|^2 \cdot |O| \cdot |S| \cdot \lceil \log_2(|t^{-1}|) \rceil \\ & & +2 * |I| \cdot |S|. \end{array}$$

In the particular case where the channel $\mathcal{P}$ is deterministic, we have that $\|P\| = |S|$ and therefore the size of the constaint matrix drops to

$$s = |I|^2 \cdot (|S| \cdot (1 + \lceil \log_2(|t^{-1}|) \rceil)) + 3 * |I| \cdot |S|.$$

The complexity is fairly high and solving the linear program directly might not be computationally feasible. In particular, the complexity is parameterised by the number of identities $|I|$ and does not exploit properties of the program channel. In the following, we show that it is possible to substantially reduce the number of parameters needed to construct an optimal channel $\mathcal{R}$.

6.2 Indistinguishable identities

Theorem 4. *Suppose that channels $\mathcal{U}$, $\mathcal{P}$ and $\mathcal{R}$ are such that*

- *identities i_1 and i_2 are indistinguishable ($i_1 \sim_{\mathcal{U}} i_2$);*
- *the channel $\mathcal{R} \otimes \mathcal{P}$ is t-private.*

Then there exists a channel $\mathcal{R}'$ such that

- *the rows of index i_1 and i_2 of $\mathcal{R}'$ are identical;*

R'	Firefox	Opera
i_0	x	$1-x$
i_1	x	$1-x$
i_2	x	$1-x$
i_3	x	$1-x$
i_4	$1-y$	y

$$max\ 4x+y \text{ s.t.} \begin{cases} x-\frac{1}{2}\cdot(4x+1-y)\leq 0 \\ 1-x-\frac{1}{2}\cdot(4-4x+y)\leq 0 \\ 1-y-\frac{1}{2}\cdot(4x+1-y)\leq 0 \\ y-\frac{1}{2}\cdot(4-4x+y)\leq 0 \\ x\geq 0 \qquad y\geq 0 \end{cases}$$

Figure 9: Randomized channel and a linear program after adding equality constraints on indistinguishable identities.

- *the channel $\mathcal{R}'\otimes\mathcal{P}$ is also t-private;*
- *and the usability of $\mathcal{R}'$ is the same as the usability of $\mathcal{R}$: $\mathbb{U}(\mathcal{R},\mathcal{U})=\mathbb{U}(\mathcal{R}',\mathcal{U})$.*

The corollary of Theorem 4 is that the linear program can be strengthened by adding equality constraints between variables. More precisely, for any pair of indistinguishable identities[4] i and i', we can add the additional constraints

$$x_{is}=x_{i's} \quad \forall s\in S.$$

The whole system of constraints can therefore be rewritten by replacing each variable x_{is} by the variable $x_{i's}$ where i' is the representative of the set $Eq_{\mathcal{U}}(i)$. This transformation has a drastic impact on the size of the resulting linear program which becomes independent from the number of identities. The number of variables is $|S|\cdot|S|$ instead of $|I|\cdot|S|$ and the number of constraints is $|S|\cdot|O|+|S|\cdot|S|+|S|$ instead of $|I|\cdot|O|+|I|\cdot|S|+|I|$.

Example 5. To demonstrate the significant simplification of the problem, in Fig. 9 we present the randomized channel $\mathcal{R}'$ and the linear program for Ex. 4, where all the indistinguishable identities get the same enforcement. One optimal solution of this problem is $x=0.9, y=0.4$, another optimal solution is $x=0.8, y=0.8$. We show these two solutions in Fig. 6 in blue color.

6.3 Greedy algorithm

We can exploit the structure of the program and reduce further the complexity at the cost of a potential non-optimal utility. In particular, we can detect identities i whose probability of guessing is below the threshold with the original channel $\mathcal{U}$. Definition 7 captures this notion.

Definition 7. Given a channel $\mathcal{P}$ and a channel $\mathcal{U}$, an identity i is *locally safe* iff

$$\frac{1}{|Eq_{\mathcal{U}\otimes\mathcal{P}}(i)|}\leq t.$$

In other words, the probability of guessing of *locally safe* identities is below the threshold t whatever the mapping of other identities.

Theorem 5. *Let Safe be the locally safe identities for P w.r.t. $\mathcal{U}$. Then there is a channel $\mathcal{R}$ such that $\mathbb{P}(\mathcal{R}\otimes\mathcal{P})\leq t$ and $\forall i\in Safe : R[i, Im(U,i)]=1$.*

[4]Definition 2 presents indistinguishable program inputs and a representative set $Eq_{\mathcal{U}}(i)$.

Proof. Given an identity $i_0 \in \mathit{Safe}$, the channel $\mathcal{R}$ can be constructed as follows:

$$\forall i \in \mathit{Safe}, R[i,j] = \begin{cases} 1 \text{ if } j = \mathit{Im}(U,i) \\ 0 \text{ if } j \neq \mathit{Im}(U,i) \end{cases}$$
$$\forall i \notin \mathit{Safe}, R[i,j] = \begin{cases} 1 \text{ if } j = \mathit{Im}(U,i_0) \\ 0 \text{ if } j \neq \mathit{Im}(U,i_0). \end{cases}$$

All locally safe identities $i \in \mathit{Safe}$ are unmodified and therefore their probability of guessing is below the threshold t. All the other identities $i \notin \mathit{Safe}$ are modified so that they are undistinguishable from the safe identity i_0 and therefore their probability of guessing is also below the threshold t. □

Using Theorem 5, we can devise a greedy algorithm for solving the linear program. Identifying the locally safe identities can be done by performing the channel composition $\mathcal{U} \otimes \mathcal{P}$ and counting the number of identical lines. Because the channel $\mathcal{U}$ is deterministic, channel composition can be performed in quadratic time. Sorting the resulting $|S|$ lines of the resulting channel and counting the number of identical lines can be done in $\mathcal{O}(|O| \cdot |S| \cdot \log(S))$. For each safe identity, i, the algorithm would assign all the variables x_{ij} such that

$$x_{ij} = \begin{cases} 1 & \text{if } j = \mathit{Im}(U,i) \\ 0 & \text{otherwise} \end{cases} .$$

The remaining constraints can be solved using standard linear programming algorithms. The size of the LP has been further reduced: there are $|S| \cdot (|S| - |\mathit{Safe}|)$ variables and $(1 + |S|) \cdot (|S| - |\mathit{Safe}|) + |S| \cdot |O|$ constraints. This is a substantial gain: the number of variables is decreasing as a linear function of $|\mathit{Safe}|$. Moreover, even if the remaining constraints do not completely vanish, they are sparser.

This algorithm is maximising usability locally because *locally safe* identities are not modified. However, there is no guarantee that this strategy would provide the maximum usability.

Example 6. Consider a threshold $t = \frac{1}{2}$ and the enforcement of $\frac{1}{2}$-privacy for the channels $\mathcal{U}$ and $\mathcal{P}$ of Fig. 10. Note that identities i_1 and i_2 are indistinguishable. As the threshold is $\frac{1}{2}$, i_1 and i_2 are locally safe

$$\mathit{Eq}_{\mathcal{U} \otimes \mathcal{P}}(i_1) = \mathit{Eq}_{\mathcal{U} \otimes \mathcal{P}}(i_2) = \{i_1, i_2\}.$$

As a result, the *greedy algorithm* is solving the following parameterised $\mathcal{R}$ channel:

R	`Firefox`	`Opera`	`Chrome`
i_1	1	0	0
i_2	1	0	0
i_3	x_{20}	x_{21}	x_{22}
i_4	x_{30}	x_{31}	x_{32}

An optimal solution with $\mathbb{U}(\mathcal{R},\mathcal{U}) = 3$ is obtained with

$$x_{20} = x_{30} = 0 \qquad x_{21} = x_{22} = x_{30} = x_{31} = x_{32} = \tfrac{1}{2}.$$

Here, the identities i_1 and i_2 keep their secrets while i_3 and i_4 are uniformly randomised over Opera and Chrome.

U	`Firefox`	`Opera`	`Chrome`
i_1	1	0	0
i_2	1	0	0
i_3	0	1	0
i_4	0	0	1

P	`A`	`B`	`C`
`Firefox`	1	0	0
`Opera`	0	1	0
`Chrome`	0	0	1

Figure 10: User channel and channel for program `if name = "Firefox" then o := A else {if name = "Opera" then o := B else o := C}`.

6.4 Decentralised algorithm

The previous algorithm can be modified to work in a completely decentralised fashion where each row of the channel R can be computed independently. If an identity i is *locally safe*, we get

$$R[i,j] = \begin{cases} 1 & \text{if } j = Im(U,i) \\ 0 & \text{otherwise} \end{cases} .$$

Otherwise, if an identity i' is not locally safe, it needs to switch configuration. This can be done by identifying an identity that is locally safe. If there is none, the identity i maximising $|Eq_{\mathcal{U}\otimes\mathcal{P}}(i)|$ can be chosen and therefore the identities are all indistinguishable.

Example 7. Continuing with the channels of Fig. 10, the decentralised algorithm also allows identities i_1 and i_2 to keep their secrets unchanged. However, i_3 and i_4 would switch to the safe identity i_1. Compared to the greedy algorithm, the decentralised algorithm obtains a smaller usability : $\mathbb{U}(\mathcal{R},\mathcal{U}) = 2$.

For deterministic programs, a quantitative information flow dynamic (or hybrid) monitor *e.g.*, [2] can evaluate if the knowledge contained in a program result is below the threshold. Using the decentralised algorithm, if the quantity of information is below the threshold, it is safe to output the result. If the identity is not proved locally safe by the dynamic monitor, the identity i maximising $|Eq_{\mathcal{U}\otimes\mathcal{P}}(i)|$ can be obtained using the previous algorithm. This identity i should be used instead of the original user identity. Note that returning a result $\perp$ distinct from existing outputs *i.e.,* an observable termination of the program does not achieve the same privacy guarantee. In practice, we can expect that there are many *locally* safe configurations and they can be identified for instance by enumerating identities by decreasing popularity and for instance running an hybrid monitor.

Using this scheme, the usability might not be optimal. Yet, the computation is decentralised and the overhead can be small for identity for which local safety can be established by hybrid monitoring.

6.5 Incremental approach

The previous approaches might require too much computing power or lack usability. An interesting trade-off can be obtained by solving the linear program incrementally. In this scenario, each time an identity i intends to run a program P, it queries a central entity to get its (randomised) secret configuration. The central entity updates the current channel $\mathcal{R}$ based on these requests. Initially, the central entity can consider the uniformly random channel[5]. For each request from an identity i, the central entity can trigger a partial optimisation specific to the secret configuration s of i. In terms of the linear program, it consists in only optimising the line of R corresponding to

[5]This channel is either t-private or none is (see Theorem 3).

the secret configuration of i. Hence, the resulting linear program has only $|S|$ variables and can solved more efficiently. Each optimisation step necessarily improves the global usability but also the usability of i. The optimisation has also a fairness property: the optimisation is biased based of the frequency of the secret configurations. However, as illustrated by the following example, the global optimal usability is not guaranteed. Moreover, upon convergence, the utility depends on the order of the optimisation steps.

Example 8. Consider the channels $\mathcal{U}$ and $\mathcal{P}$ of Fig. 10 and the uniformly random matrix R ($R[i,j] = \frac{1}{3}$ for all i, j). The initial usability is $\mathbb{U}(\mathcal{R},\mathcal{U}) = 1$.

After a request from i_1, the local optimisation for the secret Firefox yields an improved usability $\mathbb{U}(\mathcal{R},\mathcal{U}) = \frac{8}{3}$ and i_1 does not need to be randomised

$$R[i_1, \texttt{Firefox}] = 1.$$

Moreover, as i_1 and i_2 are indistinguishable, they get the same enforcement and therefore $R[i_2, Firefox] = 1$. In this configuration, neither i_3 nor i_4 can be locally optimised.

If the first optimisation request is from i_3, the local optimisation for the secret "Opera" yields an improved $\mathbb{U}(\mathcal{R},\mathcal{U}) = 2$ and i_3 does not need to be randomised

$$R[i_3, \texttt{Opera}] = 1.$$

In this configuration, neither i_1 nor i_2 can be optimised. Yet, the usability for i_4 can still be optimised. We obtain:

$$R[i_4] = [0; \frac{1}{3}; \frac{2}{3}]$$

and an improved $\mathbb{U}(\mathcal{R},\mathcal{U}) = \frac{7}{3}$. In this configuration, none of the identity can be locally optimised.

6.6 Summary of algorithms

In this part, we have presented several algorithms for synthesising a channel $\mathcal{R}$ such that $\mathcal{R} \otimes \mathcal{P}$ is t-private. All the algorithms are based on solving the Linear Program whose optimal solution yields the $\mathcal{R}$ channel with maximum usability $\mathbb{U}(\mathcal{R},\mathcal{U})$. We proved in Section 6.2 that there always exist an optimal solution such that indistinguishable identities ($i \simeq_{\mathcal{U}} i'$) have the same enforcement *i.e.*, $\mathcal{R}[i] = \mathcal{R}[i']$. We exploit this property to simplify the LP formulation and reduce the number of variables from $|I| \cdot |S|$ to $|S| \cdot |S|$.

The traditional linear programming algorithm was then refined into a series of more specialised algorithms. The *greedy algorithm* (Section 6.3) exploit further indistinguishability and ensures that identities which probability of guessing is below the threshold t do not need to switch their configuration. Computing this probability of guessing is much cheaper than solving the full Linear Program. However, the usability of the final channel $\mathcal{R}$ might not be optimal. In Section 6.4, we propose a decentralised algorithm allowing identities to make their enforcement decision in a decentralised manner. This is a refinement of the *greedy algorithm* where identities whose probability of guessing is above the threshold t can switch to an arbitrary configuration as soon as its probability of guessing is below the threshold. From the implementation point of view, the advantage is that each enforcement decision is made locally and the enforcement is transparent for identities whose probability of guessing is below the threshold. However, other identities need to systematically switch configuration, thus limiting usability. The last approach (Section 6.5) is centralised but based on an *incremental* optimisation biased towards the distribution of configurations. The enforcement is optimised on a per-configuration basis, and the usability increases as more configurations are considered. Therefore, at each optimisation step,

the linear program has $|S|$ free variables instead of $|S| \cdot |S|$. Moreover, as opposed to the decentralised algorithm, identities whose probability of guessing is above the threshold do not need to systematically switch configuration. Compared to the greedy algorithm, identities whose probability of guessing is below the threshold migth be required to switch configurations in order to optimise usability.

7 Enforcing t-privacy for any program

In this section we consider a case when the program channel $\mathcal{P}$ is unknown or cannot be computed. In this case, only the user channel $\mathcal{U} = (I, S, U)$ is given, and we need to ensure that for any program $\mathcal{P}$, the probability of identifying the user is bounded by t.

7.1 Randomisation of channel $\mathcal{U}$ with uniform distribution

To provide a generic solution for any user channel, we first propose a randomization for a user channel that models a uniform distribution of the user identities w.r.t. their configurations. We denote this channel by $\mathcal{U}_{Id} = (I, S, U_{Id})$, where U_{Id} is an identity matrix. Like in previous sections, we find a randomised channel $\mathcal{R}_{Id} = (I, I, R_{Id})$ s.t. t-privacy is enforced while usability is maximised by solving a (simplified) linear program. Notice that the threshold t must be enforceable for a channel $\mathcal{U}_{Id}$ in a sense of Theorem 3.

$$max \ \sum x_{ii} \text{ s.t.}$$
$$\begin{cases} 0 \leq x_{ij} \leq 1 & \forall i, j \in I \\ \sum_{j \in I} x_{ij} = 1 & \forall i \in I \\ x_{ik} - t \cdot \sum_{j \in I} x_{jk} \leq 0 & \forall i, k \in I \end{cases}$$

We find the following solution to this problem:

$$R_{Id}[i, j] = \begin{cases} t & \text{if } i = j \\ \frac{1-t}{|I|-1} & \text{otherwise} \end{cases}$$

with the probability of guessing: $\mathbb{P}(\mathcal{R}_{Id}) = t$ and the usability: $\mathbb{U}(\mathcal{R}_{Id}, \mathcal{U}_{Id}) = t \cdot |I|$. Interestingly, usability depends on a threshold: the higher is the threshold t, the bigger is the probability that the user can be identified, the less his original configuration should change, and hence the more usability would be provided.

7.2 Randomisation of an arbitrary channel $\mathcal{U}$

We now construct a randomised user channel $\mathcal{R}$ from a given user channel $\mathcal{U}$, that satisfies the t-privacy for any program channel $\mathcal{P}$: $\forall \mathcal{P} : \mathbb{P}(\mathcal{R} \otimes \mathcal{P}) \leq t$. Like before, a threshold t must be enforceable in a sense of Theorem 3.

We build a channel $\mathcal{R}$ by starting from a channel $\mathcal{R}_{Id}$ and merging the columns of the identities that share the same configuration in the user channel $\mathcal{U}$.

$$R[i, s] = \begin{cases} t + \frac{1-t}{|I|-1} \cdot (n_s - 1) & \text{if } i \in Pre(U, s) \\ \frac{1-t}{|I|-1} \cdot n_s & \text{otherwise} \end{cases} \tag{3}$$

where $n_s = |Pre(U, s)|$. We now prove the main properties of the constructed channel $\mathcal{R}$.

Lemma 1 (Well-formedness)**.** *Given a user channel $\mathcal{U} = (I, S, U)$, a randomized channel $\mathcal{R} = (I, S, R)$, where R is computed by equation (3) is well-formed:*

- $0 \leq R[i,s] \leq 1$ *for all* $i \in I$, $s \in S$, *meaning each parameter is a probability*
- $\sum_{s \in S} R[i,s] = 1$ *for all* $i \in I$, *meaning each line in matrix represents a distribution*

Lemma 2 (t-privacy)**.** *Given a user channel* $\mathcal{U}$*, the randomised channel* $\mathcal{R}$*, where* R *is computed by equation (3) for an enforceable threshold* t*, is* t*-private:*

$$\mathbb{P}(\mathcal{R}) \leq t.$$

We can now prove the main theorem of this section: a constructed randomized user channel $\mathcal{R}$ can ensure t-privacy for any program channel $\mathcal{P}$.

Theorem 6. *For any user channel* $\mathcal{U} = (I, S, U)$*, the randomised user channel* $\mathcal{R} = A(\mathcal{U})$ *ensures* t*-security for any program channel* $\mathcal{P} = (S, O, P)$*:*

$$\forall \mathcal{P} : \mathbb{P}(\mathcal{R} \otimes \mathcal{P}) \leq t$$

We do not prove that the randomised user channel $\mathcal{R}$ provides an optimal usability. The usability of a solution $\mathcal{R}$ given a user channel $\mathcal{U}$ is:

$$\mathbb{U}(\mathcal{R},\mathcal{U}) = \sum_{s \in S} |Pre(U,s)| \cdot (t + \frac{1-t}{|I|-1} \cdot (|Pre(U,s)| - 1)).$$

We now state the lower and upper bounds of the usability that is defined by the characteristics of the channel $\mathcal{U}$:

$$\mathbb{U}(\mathcal{R},\mathcal{U}) \geq l \cdot |S| \cdot (t + \frac{1-t}{|I|-1} \cdot (l-1)) \geq t \cdot |I|$$

$$\mathbb{U}(\mathcal{R},\mathcal{U}) \leq h \cdot |S| \cdot (t + \frac{1-t}{|I|-1} \cdot (h-1)) \leq |I|$$

where $l = \min_{s \in S} Pre(U,s)$ and $h = \max_{s \in S} Pre(U,s)$.

In a general case, the randomized user channel $\mathcal{R}$ constructed in this section, will provide a solution with a reduced usability with respect to the solutions provided by other approaches in a previous section. The reason for this is the fact that a program channel $\mathcal{P}$ may already make some of the configurations indistinguishable thus the users of such configurations could obtain a better usability.

8 Related work and Discussion

8.1 Evaluation of information flow

Mardziel *et al.* [20, 21] define the notion of *threshold secure* program. This is a generalisation of t-privacy allowing to attach different thresholds to different secrets. In our context, as we wish to protect privacy (and not secrets), only a single threshold is needed. Mardziel *et al.* are using an abstract domain of probabilistic polyhedra for computing an over-approximation of the threshold security of a program. They exploit this information to implement a simple enforcement algorithm: If the program is *threshold secure*, it runs without modification; if it is not *threshold secure*, it does not run at all. Our enforcement offers a better usability at the price of randomising the program inputs. Yet, our enforcement algorithms can ensure a minimum randomisation that is thus affecting a minimal set of users. For example, a greedy algorithm ensures that locally safe users will always run the program without any changes.

Klebanov [15, 16] has proposed efficient algorithms for exactly computing standard quantitative information flow measures of programs such as conditional (minimal) guessing entropy. The algorithms are either based on SAT-solving techniques [16] or on *extended Barvinok counting* [27]. These techniques are applied only to restricted classes of programs. SAT-based techniques require a propositional encoding of programs. Extended Barvinok counting consists in computing the number of integer points in a parametrised polyhedron and thus applies to programs that can be specified using linear integer arithmetic. In our theoretical setting, channels can model arbitrary terminating programs with a finite input/output relation but constructing explicitly the channel matrix could be costly. More efficient enforcement algorithms could certainly benefit from syntactic program restrictions. For deterministic programs, Klebanov's approaches can be adapted for deciding whether a program is t-private. Notice that Klebanov is only concerned with the problem of quantifying information flows and does not consider enforcement. However, this information can be directly exploited to implement the simple enforcement proposed by Mardziel *et al.*.

Köpf and Rybalchenko [17] approximate the entropy of a program using a combination of static and dynamic analyses, where random sampling is enhanced by a symbolic backward analysis. The method ensures a rapid convergence rate but the guarantee is probabilistic. Because t-privacy is a property quantifying over all the program inputs, it cannot be established by random sampling. Moreover, the purpose of t-privacy is to protect all users (especially those whose configurations are rare).

8.2 Database privacy definitions

8.2.1 k-anonymity

Sweeney [26] proposes *k-anonymity*, a security property of anonymization services that requires that every individual is anonymous within some set of at least size k. k-anonymity was not widely adopted as a privacy definition for two major reasons: 1) the values of sensitive attributes in the remaining set could be discovered due to their little diversity; 2) attackers with background knowledge of the distribution of the secret inputs can still infer some information about the secrets despite the fact that k-anonymity is enforced.

The first problem related to k-anonymity is shown by an example when a program's output could have been caused by one of k possible inputs, but one of those inputs is much more probable than the rest. After observing a k-anonymous answer of a query, the post distribution of the secrets represents this knowledge of the attacker. The probability of guessing the secret given this knowledge is bounded by t thanks to t-privacy guarantee. The second problem is not applicable in our case since the attacker does not have any background knowledge: he collects all the data about the user through the execution of the program, and he has no history of interaction with the user because he cannot identify the user.

To propose the solution to the above-mentioned issues related to the definition of k-anonymity, Machanavajjhala et al. [19] proposed a notion of l-diversity to specify that the remaining sensitive data must be diverse. This definition does not apply to our case.

8.2.2 Differential privacy

Dwork et al. [10] proposed a new privacy definition: a query to the database is *ϵ-differentially private* if and only if its answer is very similar to this query answer over a database that differs in only one record. In the other words, one record in a database does not significantly change the answer of a query. Differential privacy was designed to reason about databases, while our probability of guessing is defined over guessing the only one secret: the user identity. Mardziel

et al., [21] make the observation that threshold based privacy and differential privacy can by formally compared using the notion of ϵ-*adversarial privacy* [23]. This notion was proven to be equivalent to differential privacy for a certain class of attacker's a priori distribution of input secrets (uniform distribution in our case). In our notations ϵ-adversarial privacy can be defined as follows.

Definition 8. A channel $\mathcal{C} = (C, I, O)$ is ϵ-*adversarially private* iff for all input secrets $i \in I$ and for all output $o \in O$ we have $p(i|o) \leq e^{\epsilon} p(i)$.

As we consider only one a priori distribution of secrets, ϵ-*adversarial privacy* definition coincides with our definition of t-*privacy* where $t = \frac{e^{\epsilon}}{|I|}$. Because differential privacy protects against a class of attacker's the security guarantee is formally stronger. For this reason, algorithms for differential privacy [11] would therefore randomise scripts that are already t-private (but not ϵ-*adversarial private*) thus reducing their usability. In our fingerprinting context, we exploit the attacker's a priori knowledge for synthesising a channel that is t-private with minimal randomisation.

9 Conclusions and further work

Web tracking uses browser fingerprinting to identify users via scripts that obtain information about the browser's configuration. To protect users from such tracking, we propose a privacy enforcement mechanism based on randomisation of the script input. We have formalized this idea through the notion of t-privacy which guarantees that the probability of guessing an identity by observing a script output is below a threshold t. We have presented a series of algorithms for enforcing t-privacy, all based on a Linear Programming formulation of the problem. The algorithms provide various trade-off between efficiency and *usability* where usability means that as little randomization as possible is used. The exact resolution of the LP provides optimal usability. We also provide an enforcement mechanism ensuring the t-privacy of arbitrary programs at the cost of a reduced usability.

In our model, both the attacker and the enforcement mechanism have perfect knowledge of the channel $\mathcal{U}$ *i.e.*, a perfect knowledge of the distribution of the configurations. In a fingerprinting context, there are databases providing detailed information about the statistics of browser configurations [12] but a perfect, real-time knowledge of the distribution of the browser configuration worldwide is not realistic. As future work, we will investigate how to extend our framework to model partial knowledge *e.g.,* the fact that the user channel $\mathcal{U}$ belongs to a set $\mathbf{U}$.

One extension could *e.g.*, consist in synthesising a channel $\mathcal{R}$ that would ensure t-privacy respect to the set $\mathbf{U}$. If $\mathbf{U}$ is expressed as a parameterised distribution, the enforcement problem can be stated as a non-linear optimisation problem instead of a Linear Program. Another extension consists in considering that the attacker might have an imprecise pre-belief [7] ($\mathcal{U} \in \mathbf{U}$) and ensure, for instance, that the channel is t-private with high probability. We could also consider that the attacker does not know the precise $\mathcal{R}$ channel but only the properties it enforces. In that case, the enforcement could ensure t-privacy at a better usability.

A longer-term extension of our model consists in modelling the dynamics of browser configurations. This dynamics is an obstacle to fingerprinting as fingerprints need to be resist the modification of configurations. In theory, this should allow to design enforcement mechanisms providing a better usability. One of the author has started a preliminary practical evaluation of the evolution of fingerprints [3]. However, more research is needed to incorporate this knowledge into a formal model.

References

[1] Gunes Acar, Marc Juárez, Nick Nikiforakis, Claudia Díaz, Seda F. Gürses, Frank Piessens, and Bart Preneel. FPDetective: dusting the web for fingerprinters. In ACM, editor, *CCS*, pages 1129–1140, 2013.

[2] Frédéric Besson, Nataliia Bielova, and Thomas Jensen. Hybrid information flow monitoring against web tracking. In IEEE, editor, *CSF'13*, pages 240–254, 2013.

[3] Nataliia Bielova and Patricio Palladino. Stopfingerprinting, 2013. `https://stopfingerprinting.inria.fr/`.

[4] K. Boda. Firegloves.

[5] Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Prakash Panangaden. Probability of error in information-hiding protocols. In IEEE, editor, *CSF'07*, pages 341–354, 2007.

[6] Kostas Chatzikokolakis, Catuscia Palamidessi, Geoffrey Smith, et al. Measuring information leakage using generalized gain functions. In *IEEE Computer Security Foundations Symposium (CSF'12)*, pages 265–279. IEEE, 2012.

[7] Michael R. Clarkson, Andrew C. Myers, and Fred B. Schneider. Quantifying information flow with beliefs. *Journal of Computer Security*, 17(5):655–701, 2009.

[8] T. M. Cover and J. A. Thomas. *Elements of Information Theory (2. ed.)*. Wiley, 2006.

[9] Charles A. Desoer. *Communication through channels in cascade*. PhD thesis, Massachusetts Institute of Technology, 1953.

[10] Cynthia Dwork. Differential privacy. In *ICALP*, volume 4052 of *LNCS*, pages 1–12. Springer, 2006.

[11] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *TCC 2006*, volume 3876 of *LNCS*, pages 265–284. Springer, 2006.

[12] P. Eckersley. The Panopticlick project. `https://panopticlick.eff.org`.

[13] Barbara Espinoza and Geoffrey Smith. Min-entropy as a resource. *Inf. Comp.*, 226:57–75, 2013.

[14] Friedrich Gretz, Joost-Pieter Katoen, and Annabelle McIver. Operational versus weakest precondition semantics for the probabilistic guarded command language. In *QEST*, pages 168–177. IEEE, 2012.

[15] Vladimir Klebanov. Precise quantitative information flow analysis using symbolic model counting. In *QASA*, 2012.

[16] Vladimir Klebanov, Norbert Manthey, and Christian Muise. SAT-based analysis and quantification of information flow in programs. In *Proc. of 10th International Conference on Quantitative Evaluation of Systems (QUEST)*, volume 8054 of *LNCS*, pages 156–171. Springer, 2013.

[17] Boris Köpf and Andrey Rybalchenko. Approximation and randomization for quantitative information-flow analysis. In *CSF'10*, pages 3–14. IEEE, 2010.

[18] Boris Köpf and Geoffrey Smith. Vulnerability bounds and leakage resilience of blinded cryptography under timing attacks. In *IEEE Computer Security Foundations Symposium (CSF'10)*, pages 44–56, 2010.

[19] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. *ACM Trans. Knowl. Discov. Data*, 1(1), 2007.

[20] Piotr Mardziel, Stephen Magill, Michael Hicks, and Mudhakar Srivatsa. Dynamic Enforcement of Knowledge-based Security Policies. In IEEE, editor, *CSF'11*, pages 114–128, 2011.

[21] Piotr Mardziel, Stephen Magill, Michael Hicks, and Mudhakar Srivatsa. Dynamic enforcement of knowledge-based security policies using probabilistic abstract interpretation. *Journal of Computer Security*, 21(4):463–532, 2013.

[22] Nick Nikiforakis, Alexandros Kapravelos, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. In *IEEE Symposium on Security and Privacy*, pages 541–555, 2013.

[23] Vibhor Rastogi, Michael Hay, Gerome Miklau, and Dan Suciu. Relationship privacy: Output perturbation for queries with joins. In *PODS '09*, pages 107–116. ACM, 2009.

[24] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, 1998.

[25] Geoffrey Smith. On the Foundations of Quantitative Information Flow. In *Foundations of Software Science and Computational Structures*, volume 5504 of *LNCS*, pages 288–302. Springer, 2009.

[26] Latanya Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):571–588, 2002.

[27] Sven Verdoolaege, Rachid Seghir, Kristof Beyls, Vincent Loechner, and Maurice Bruynooghe. Counting integer points in parametric polytopes using barvinok's rational functions. *Algorithmica*, 48(1):37–66, 2007.

The appendix contains the proofs for the theorems of the paper.

Theorem 1. *For any channel $\mathcal{C} = (I, O, C)$, the probability of guessing the input is bounded:* $\mathbb{P}(\mathcal{C}) \geq \frac{1}{|I|}$.

Proof. Following the definition of probability of guessing, $\mathbb{P}(\mathcal{C}) = \max_{i \in I, o \in O} p(i|o)$. Let's denote an output o' that maximizes the probability $p(i|o)$, so that

$$\mathbb{P}(\mathcal{C}) = \max_{i \in I} p(i|o')$$

Given such output o', $p(i|o')$ forms a posterior distribution, and therefore $\sum_{i \in I} p(i|o') = 1$. Hence, one of the two statements hold:

- either $\forall i \in I : p(i|o') = \frac{1}{|I|}$, hence $\mathbb{P}(\mathcal{C}) = \frac{1}{|I|}$,
- or $\exists i \in I : p(i|o') > \frac{1}{|I|}$, hence $\mathbb{P}(\mathcal{C}) > \frac{1}{|I|}$.

□

Theorem 2. *The probability of guessing the identity from a user channel $\mathcal{U}$ does not increase if a program channel $\mathcal{P}$ is added:* $\mathbb{P}(\mathcal{U} \otimes \mathcal{P}) \leq \mathbb{P}(\mathcal{U})$.

Proof. Given a user channel $\mathcal{U} = (I, S, U)$, a program channel $\mathcal{P} = (S, O, P)$, and a cascade of channels $\mathcal{U} \otimes \mathcal{P} = (I, O, p(o|i))$, let us rewrite the definition of probability of guessing as a maximum probability of an a posteriori distribution:

$$\max_{i \in I, o \in O} p(i|o) \leq \max_{i \in I, s \in S} p(i|s). \tag{4}$$

We rewrite the definition of the probability $p(i|o)$:

$$\begin{aligned}
p(i|o) &= \frac{p(i,o)}{p(o)} = \frac{p(o|i) \cdot p(i)}{p(o)} \\
&= \frac{\sum_{s \in S} p(i) \cdot U[i,s] \cdot P[s,o]}{\sum_{j \in I} \sum_{s \in S} p(j) \cdot U[j,s] \cdot P[s,o]} \\
&= \frac{\sum_{s \in S} p(i) \cdot p(s|i) \cdot P[s,o]}{\sum_{s \in S} \sum_{j \in I} p(j) \cdot p(s|j) \cdot P[s,o]} \\
&= \frac{\sum_{s \in S} p(i,s) \cdot P[s,o]}{\sum_{s \in S} p(s) \cdot P[s,o]} \\
&= \frac{\sum_{s \in S} p(s) \cdot p(i|s) \cdot P[s,o]}{\sum_{s \in S} p(s) \cdot P[s,o]}
\end{aligned}$$

For readability by $f(s,o)$ we denote $p(s) \cdot P[s,o]$. To prove the statement (4), we have to prove that $\forall i, o : p(i|o) \leq p(i', s')$ where i' and s' maximize $p(i|s)$. Therefore, we need to prove that for all i, o:

$$\frac{\sum_{s \in S} p(i|s) \cdot f(s,o)}{\sum_{s \in S} f(s,o)} \leq p(i'|s')$$

which holds since $p(i,s) \leq p(i', s')$ for all i, s. □

Theorem 3 (Enforceability)**.** *A threshold t is enforceable for a user channel $\mathcal{U} = (I, S, U)$ and any program channel $\mathcal{P}$ if and only if $\frac{1}{|I|} \leq t$.*

Proof. It-direction: If t is enforceable for $\mathcal{U}$ and $\mathcal{P}$, then according to Definition 5, there exists a randomized channel $\mathcal{R}$, such that $\mathbb{P}(\mathcal{R} \otimes \mathcal{P}) \leq t$. Given that $\mathcal{R} \otimes \mathcal{P}$ has the same number of inputs $|I|$ as $\mathcal{U}$, and following Theorem 1, we have $t \geq \mathbb{P}(\mathcal{R} \otimes \mathcal{P}) \geq \frac{1}{|I|}$.

Only-if direction: There exists a fully randomized channel $\mathcal{R} = (I, S, R)$, where $R[i, s] = \frac{1}{|S|}$ for all i, s. By construction,

$$\mathbb{P}(\mathcal{R}) = \max_{i,s} \frac{R[i,s]}{\sum_{j \in I} R[j,s]} = \frac{\frac{1}{|S|}}{|I| \cdot \frac{1}{|S|}} = \frac{1}{|I|}.$$

Following Theorem 2 and using the *only-if* condition stating $t \geq \frac{1}{|I|}$, we have:

$$\mathbb{P}(\mathcal{R} \otimes \mathcal{P}) \leq \mathbb{P}(\mathcal{R}) = \frac{1}{|I|} \leq t.$$

Therefore, t is enforceable for a user channel $\mathcal{U} = (I, S, U)$ and any program channel $\mathcal{P}$. □

Theorem 4. *Suppose that channels $\mathcal{U}$, $\mathcal{P}$ and $\mathcal{R}$ are such that*

- *identities i_1 and i_2 are indistinguishable ($i_1 \sim_{\mathcal{U}} i_2$);*
- *the channel $\mathcal{R} \otimes \mathcal{P}$ is t-private.*

Then there exists a channel $\mathcal{R}'$ such that

- *the rows of index i_1 and i_2 of $\mathcal{R}'$ are identical;*
- *the channel $\mathcal{R}' \otimes \mathcal{P}$ is also t-private;*
- *and the usability of $\mathcal{R}'$ is the same as the usability of $\mathcal{R}$: $\mathbb{U}(\mathcal{R},\mathcal{U}) = \mathbb{U}(\mathcal{R}',\mathcal{U})$.*

Proof. Without loss of generality, consider that i_1 and i_2 are the first two rows of R (*i.e.,* $R = [R[i_1]; R[i_2]; R|_{I \setminus \{i_1,i_2\}}]$) and that $Im(R, i_1) = Im(R, i_2) = s_1$ for some s_1. The channel $\mathcal{R}'$ can be constructed from $\mathcal{R}$ by averaging the rows for i_1 and i_2. Let $a = \frac{1}{2}(R[i_1] + R[i_2])$, we have

$$R' = [a; a; R|_{I \setminus \{i_1,i_2\}}]$$

We have $R'[i_1] = R'[i_2] = a$ and the rows are therfore identical. We have $\mathbb{U}(\mathcal{R}',\mathcal{U}) = (\mathbb{U}(\mathcal{R},\mathcal{U}) - R[i_1, s] - R[i_2, s]) + 2 \times a[s]$. As $a[s] = \frac{1}{2}(R[i_1, s_1] + R[i_2, s_1])$, the usability is therefore unchanged. To prove that the channel $\mathcal{R}' \otimes \mathcal{P}$ is also t-private, it remains to prove (for all i and o) that

$$\sum_{s \in S} R'[i,s] \cdot P[s,o] - t \cdot \sum_{j \in I} \sum_{s \in S} R'[j,s] \cdot P[s,o] \leq 0$$

By definition of R' and the fact that $R'[i_1, s] + R'[i_2, s] = 2 * a[s] = R[i_1, s] + R[i_2, s]$, the constraint is equivalent to

$$\sum_{s \in S} R'[i,s] \cdot P[s,o] - t \cdot \sum_{j \in I} \sum_{s \in S} R[j,s] \cdot P[s,o] \leq 0$$

If $i \notin \{i_1; i_2\}$, we have $\sum_{s \in S} R'[i,s] \cdot P[s,o] = \sum_{s \in S} R[i,s] \cdot P[s,o]$ and the constraint holds because $\mathcal{R} \otimes \mathcal{P}$ is t-private. If $i = i_1, i_2$, we have $\sum_{s \in S} R'[i,s] \cdot P[s,o] = \sum_{s \in S} \frac{1}{2}(R[i_1, s] +$

$R[i_2, s]) \cdot P[s, o]$. W.l.o.g., suppose that $\sum_{s\in S} R[i_1, s] \cdot P[s, o] \leq \sum_{s\in S} R[i_2, s] \cdot P[s, o]$. It follows that $\sum_{s\in S} R'[i, s] \cdot P[s, o] \leq \sum_{s\in S} R[i_2, s] \cdot P[s, o]$ and we conclude

$$\begin{array}{l}\sum_{s\in S} R'[i, s] \cdot P[s, o] - t \cdot \sum_{j\in I} \sum_{s\in S} R[j, s] \cdot P[s, o] \leq \\ \sum_{s\in S} R[i_2, s] \cdot P[s, o] - t \cdot \sum_{j\in I} \sum_{s\in S} R[j, s] \cdot P[s, o] \leq 0\end{array}$$

Hence, the constraints hold. The channel $\mathcal{R}' \otimes \mathcal{P}$ is therefore t-private and this concludes the proof. □

Theorem 5. *Let Safe be the locally safe identities for P w.r.t. $\mathcal{U}$. Then there is a channel $\mathcal{R}$ such that $\mathbb{P}(\mathcal{R} \otimes \mathcal{P}) \leq t$ and $\forall i \in Safe : R[i, Im(U, i)] = 1$.*

Proof. Given an identity $i_0 \in Safe$, the channel $\mathcal{R}$ can be constructed as follows:

$$\begin{array}{l}\forall i \in Safe, R[i, j] = \left\{\begin{array}{l} 1 \text{ if } j = Im(U, i) \\ 0 \text{ if } j \neq Im(U, i)\end{array}\right. \\ \forall i \notin Safe, R[i, j] = \left\{\begin{array}{l} 1 \text{ if } j = Im(U, i_0) \\ 0 \text{ if } j \neq Im(U, i_0)\end{array}\right.\end{array}$$

All locally safe identities $i \in Safe$ are unmodified and therefore their probability of guessing is below the threshold t. All the other identities $i \notin Safe$ are modified so that they are undistinguishable from the safe identity i_0 and therefore their probability of guessing is also below the threshold t. □

Lemma 1 (Well-formedness)**.** *Given a user channel $\mathcal{U} = (I, S, U)$, a randomized channel $\mathcal{R} = (I, S, R)$, where R is computed by equation (3) is well-formed:*

- $0 \leq R[i, s] \leq 1$ *for all $i \in I$, $s \in S$, meaning each parameter is a probability*
- $\sum_{s\in S} R[i, s] = 1$ *for all $i \in I$, meaning each line in matrix represents a distribution*

Proof. For simplicity of reading, by N we denote $|I|$. Proving $0 \leq R[i, s] \leq 1$ for all $i \in I$, $s \in S$:

- If $i \in Pre(U, s)$ then
$$R[i, s] = t + \frac{1-t}{N-1} \cdot (|Pre(U, s)| - 1).$$
Given that $1 \leq |Pre(U, s)| \leq N$, we have:
$$0 < t \leq R[i, s] \leq t + 1 - t = 1.$$
- If $i \notin Pre(U, s)$ then
$$R[i, s] = \frac{1-t}{N-1} \cdot |Pre(U, s)|.$$
Given that $1 \leq |Pre(U, s)| \leq N - 1$, we have:
$$0 < \frac{1-t}{N-1} \leq R[i, s] \leq 1 - t < 1.$$

Proving $\sum_{s\in S} R[i, s] = 1$ for all $i \in I$. Let's take an arbitrary row i of a matrix R and split all the secrets s in two sums: $s \in Im(U, i)$ and $s \notin Im(U, i)$. Since $\mathcal{U}$ is a deterministic channel, for every user i there is only one s such that $U[i, s] = 1$, and therefore $|Im(U, i)| = 1$. For our row i, let's denote this single secret by s_0. Notice that $s \in Im(U, i)$ means that $i \in Pre(U, s)$. Hence,

$$\begin{aligned}
\textstyle\sum_{s\in S} R[i,s] &= \textstyle\sum_{s\in Im(U,i)} t + \frac{1-t}{N-1}\cdot(|Pre(U,s)|-1) \\
&\quad + \textstyle\sum_{s\notin Im(U,i)} \frac{1-t}{N-1}\cdot|Pre(U,s)| \\
&= t + \tfrac{1-t}{N-1}\cdot|Pre(U,s_0)| - \tfrac{1-t}{N-1} \\
&\quad + \textstyle\sum_{s\in S\setminus\{s_0\}} \frac{1-t}{N-1}\cdot|Pre(U,s)| \\
&= t - \tfrac{1-t}{N-1} + \tfrac{1-t}{N-1}\cdot\textstyle\sum_{s\in S}|Pre(U,s)| \\
&= t - \tfrac{1-t}{N-1} + \tfrac{1-t}{N-1}\cdot N = t+1-t = 1
\end{aligned}$$

□

Lemma 2 (t-privacy)**.** *Given a user channel $\mathcal{U}$, the randomised channel $\mathcal{R}$, where R is computed by equation (3) for an enforceable threshold t, is t-private:*

$$\mathbb{P}(\mathcal{R}) \leq t.$$

Proof. The probability of guessing the input of a channel $\mathcal{R}$ is:

$$\mathbb{P}(\mathcal{R}) = \max_{i\in I, s\in S} \frac{R[i,s]}{\sum_{i'\in I} R[i',s]}.$$

To prove the lemma, let us prove that for any fixed i and s:

$$R[i,s] \leq t\cdot\sum_{i'\in I} R[i',s].$$

We first compute the sum on the right-hand side for s fixed s (here $n_s = |Pre(U,s)|$):

$$\begin{aligned}
\textstyle\sum_{i'\in I} R[i',s] &= \textstyle\sum_{i\in Pre(U,s)}(t + \frac{1-t}{N-1}\cdot(n_s-1)) \\
&\quad + \textstyle\sum_{i\notin Pre(U,s)} \frac{1-t}{N-1}\cdot n_s \\
&= (t + \tfrac{1-t}{N-1}\cdot(n_s-1))\cdot n_s \\
&\quad + \tfrac{1-t}{N-1}\cdot n_s\cdot(N-n_s) \\
&= \tfrac{n_s}{N-1}(t\cdot N - t + n_s - 1 - t\cdot n_s + t \\
&\quad + N - n_s - t\cdot N + t\cdot n_s) \\
&= \tfrac{n_s}{N-1}\cdot(N-1) = n_s
\end{aligned}$$

There are two cases for a given fixed i:

- If $i \in Pre(U,s)$, then $R[i,s] = t + \frac{1-t}{N-1}\cdot(n_s-1)$. We now prove that $R[i,s] - t\cdot\sum_{i'\in I} R[i',s] \leq 0$, we rewrite it as:

$$\begin{aligned}
& t + \tfrac{1-t}{N-1}\cdot(n_s-1) - t\cdot n_s \\
= \;& t + \tfrac{1-t}{N-1}\cdot(n_s-1) - t\cdot(n_s-1) - t \\
= \;& \textstyle\sum_{k=1}^{n_s-1}(\frac{1-t}{N-1} - t) \leq 0
\end{aligned}$$

  because each element of the sum $\frac{1-t}{N-1} - t \leq 0$ since $t \geq \frac{1}{N}$ according to Theorem 3 on the enforceability of t.

- If $i \notin \mathit{Pre}(U, s)$, then given that t is enforceable and according to Theorem 3, $t \geq \frac{1}{N}$, we have $\frac{1-t}{N-1} \leq t$, and therefore $\frac{1-t}{N-1} \cdot n_s \leq t \cdot n_s$, meaning that $R[i, s] \leq t \cdot \sum_{i' \in I} R[i', s]$.

□

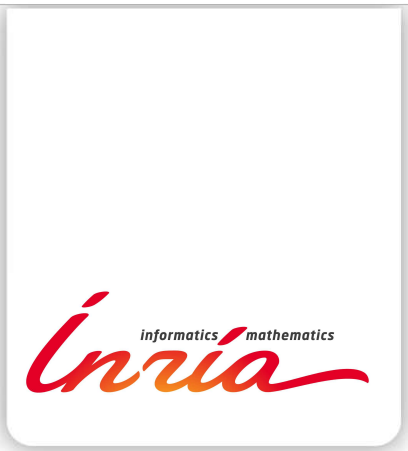
Theorem 6. *For any user channel $\mathcal{U} = (I, S, U)$, the randomised user channel $\mathcal{R} = A(\mathcal{U})$ ensures t-security for any program channel $\mathcal{P} = (S, O, P)$:*

$$\forall \mathcal{P} : \mathbb{P}(\mathcal{R} \otimes \mathcal{P}) \leq t$$

Proof. By Theorem 2, we have

$$\forall \mathcal{P} : \mathbb{P}(\mathcal{R} \otimes \mathcal{P}) \leq \mathbb{P}(\mathcal{R})$$

and according to Lemma 2: $\mathbb{P}(\mathcal{R}) \leq t$. □



RESEARCH CENTRE
RENNES – BRETAGNE ATLANTIQUE

Campus universitaire de Beaulieu
35042 Rennes Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399